```java
import java.io.*;


class SimpleShell
{

    //displays history of commands from an array list
    private static void historyOutput(ArrayList<String> history)
    {
        for(int i = 0; i < history.size(); i++)
            System.out.println(i + "    " + history.get(i));
    }

    //returns the previous command from the array list
    private static String returnPrevious(ArrayList<String> history)
    {
        return history.get(history.size() - 2);
    }

    public static void main(String[] args) throws IOException
    {
        String commandLine;
        BufferedReader console = new BufferedReader(new InputStreamReader(System.in));
        ArrayList<String> historyList = new ArrayList<String>();

        while(true)
        {

            //read what the user has entered
            System.out.println("jsh>");
            commandLine = console.readLine();

            //Creates a history Array List storing all the commands inserted by the user so
far
            historyList.add(commandLine);

            int commNum;

            //CHECKS FOR PART 3 COMMANDS
            //If user input is "history" it displays all previous user inputs
            if(commandLine.equals("history"))
            {
                historyOutput(historyList);
                //System.out.println("The size is: " + historyList.size());
            }
            else
                //command to return the previous command from the user
                if(commandLine.equals("!!") && historyList.size() >= 1)
                {
                    commandLine = returnPrevious(historyList);
                }
                else
                    //If there is no previous command to return to it displays an error
                    if(commandLine.equals("!!") && historyList.size() < 1)
                    {
                        System.out.println("Unable to run previous command for there is no
previous command.");
```

```java
                        continue;
                    }
                    else
                        //command to go to a specific 'i'th command
                        if(commandLine.charAt(0) == '!' && commandLine.length() == 2)
                        {
                            commNum = Integer.parseInt(commandLine.substring(1));
                            commandLine = historyList.get(commNum - 1);
                        }



            //if the user has entered a return, just loop again
            if (commandLine.equals(""))
                continue;

            //It separates the words of the user inputs on every "space" there is
            String[] parameterArray = commandLine.split(" ");
            System.out.println("Output String Array: " + Arrays.toString(parameterArray));

            //System.out.println("Value at index 1 of array:" +  parameterArray[1]);

            //It adds all the elements of the parameter String Array into a String Array List
            List<String> parameterList = Arrays.asList(parameterArray);




            //System.out.println("user input: " + commandLine);

            //System.out.println(System.getProperty("User current location is: " + "user.dir"));

            try
            {

                //created a ProcessBuilder objected passing the parameterList to the constructor
                ProcessBuilder pb = new ProcessBuilder(parameterList);
                System.out.println("The String Array List Command Was: " + pb.command() + "\n");

                //System.out.println(System.getProperty("User current location is: " + "user.dir"));


                //CHECKS FOR PART 2 COMMANDS
                if (commandLine.equals("cd") || commandLine.equals("cd ~"))
                    pb.directory(new File(System.getProperty("user.home")));
                else
                    if(commandLine.equals("cd /"))
                        pb.directory(new File("/")); // Go to root directory case
                    else
                        //Checks if first argument is "cd" and second argument does not contains a '/'
                        if((parameterArray[0].equals("cd")) && (parameterArray[1].indexOf('/') < 0))
                        {
                            //calls the word line after the "cd", e.g. cd music would set directory to music
                            pb.directory(new File(System.getProperty("user.dir") +"/" + parameterArray[1]));

                            //System.out.println("Parameter with no slash" + ": is " + parameterArray[1]);
                        }
                        else
                            //Checks if first argument is "cd" and second argument contains a '/'
                            if((parameterArray[0].equals("cd")) && (parameterArray[1].indexOf('/') >= 0))
                            {
                                System.out.println(System.getProperty("user.dir") + "/" + parameterArray[1]);

                                pb.directory(new File(parameterArray[1]));

                            }


                //System.out.println("The file Path was: " + filePath);


                //starts the process
                Process process = pb.start();


                //obtains output stream
                InputStream is = process.getInputStream();
                InputStreamReader isr = new InputStreamReader(is);
                BufferedReader br = new BufferedReader(isr);

                //outputs the contents returned by the command
                String line;
                while ( (line = br.readLine()) != null)
                    System.out.println(line);

                br.close();

            }
            //If invalid command, it displays the error message and waits for further commands from the user.
            catch(IOException ex)
            {

                System.out.println("\n"+ ex.toString());
                System.out.println("Possible invalid command. Please try again..." + "\n");
                continue;
            }
```

```


          }


     }

}

```